Team CASE and the 2007 DARPA Urban Challenge

Wyatt Newman, Team Lead

Case Western Reserve University
10900 Euclid Ave., Cleveland, OH 44106

wsn@case.edu

http://urbanchallenge.case.edu

1. *INTRODUCTION*

Team Case is responding to the 2007 DARPA Grand Challenge, the "Urban Challenge," with a vehicle that is both rugged and intelligent. Our vehicle, "DEXTER," designed and raced by Team ENSCO in 2005, placed 6th in the 2005 Desert Challenge. The Case/ENSCO collaboration was initiated at DARPA's networking event during the initial Urban Challenge participant's conference. ENSCO's proven vehicle and robust engineering talent augments Case's research and technology depth in biologically inspired robot design and control.

Team Case is comprised of over 50 members, with active contributors including professors, graduate students, undergraduates, and collaborators from the original Team ENSCO. Our team members are drawn from academic programs of electrical engineering, mechanical engineering, biomedical engineering, computer engineering, computer science and systems engineering. In addition to enjoying use of ENSCO's DEXTER, we are supported by donations from corporations, foundations, individuals, Case Western Reserve University and the Case Alumni Association. Additional technical contributions are being provided by Argon ST, the Air Force Institute of Technology (AFIT), and IEC Infrared Systems, Inc.

The primary challenge of this competition will be the ability to interpret imperfect data and execute context-appropriate actions in complex and ambiguous situations. Examples include: maintaining safe vehicle control in areas of GPS outage and/or obscured aerial maps; recognizing the existence and intent of other vehicles and interacting with them safely; and constantly re-evaluating the viability of current action choices vs. available contingency options. We have based our system on an architecture that will accommodate continuous growth and change because the sensor systems, planning and decision-making algorithms, and targeted expert behaviors will continue to evolve for years to come. We have thus based our system on an architecture that will accommodate continuous growth and change.

This report is organized as follows. A system overview is presented first, in terms of the system architecture. This is followed by a bottom-up presentation, including: the mechanical system and low-level controls; the physical sensor systems; the software perceptual systems; the behavioral and lower-level decision systems; and the higher-level planning and learning systems. We conclude with a description of our whole-system testing process, including simulation testing and field testing and summarize progress to date and planned developments leading to the final race.

2. *SYSTEM ARCHITECTURE*

Figure 1 shows a high-level view of our system organization. At this level of resolution, the modules are the following. The vehicle block includes DEXTER's mechanics, safety systems, servo controls and interface layer to higher-level commands. The sensor block includes the physical sensor systems and low-level associated software required for interfacing and signal preprocessing. The "observers" component incorporates a growing number of perceptual processing modules, each of which is responsible for interpreting lower-level sensory data in terms of targeted perceptual needs (e.g., localization, vehicle tracking, obstacle detection, lane sensing, etc.). The route planner is responsible for determining high-level plans (sequences of lanes to follow) that lead to successful mission completion.

The "strategy" component consists of a collection of targeted competencies implemented as finite-state automata (FSA). To distinguish these reactive FSA from strategies at higher levels of abstraction, we refer to our low-level FSA strategies as "Moods." For example, the "follow lane" mood uses information from the Lane Observer to compute appropriate vehicle motion commands. At the same time, this mood monitors relevant Observers for necessary reactions, e.g. in response to unexpected obstacles or impending collisions with other vehicles. Moods are intentionally limited in scope and are designed to perform a specific subtask well. At any instant, one and only one mood is active.

Situational uncertainty beyond the competence of an active mood induces an appeal to a higher level—the mood selector. The mood selector is responsible for evaluating information from situational Observers in the context of the current subgoal. The Mood Selector is responsible for choosing, parameterizing, initializing and activating the mood that is most appropriate in context.

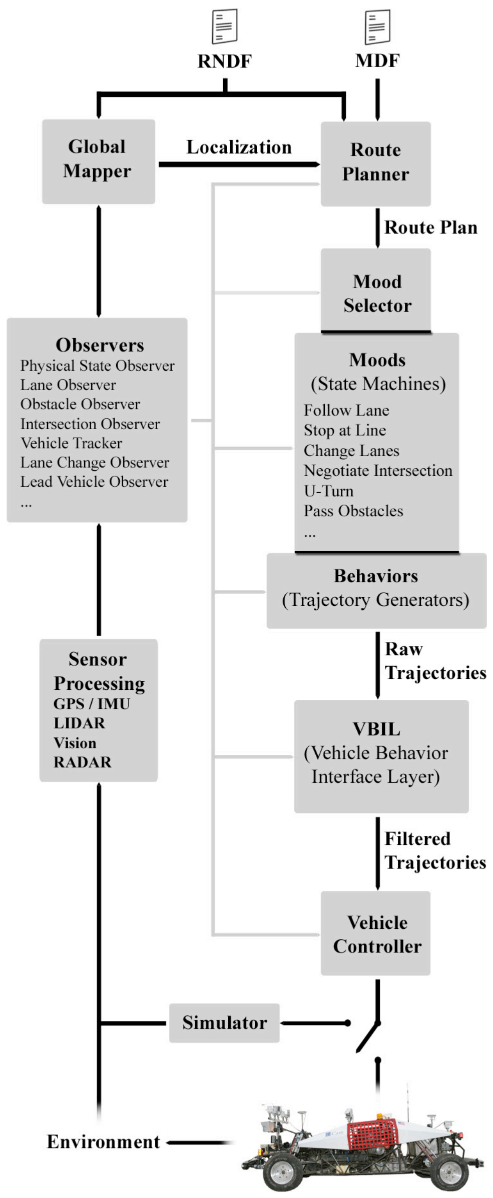


Figure 1: Block diagram of Team Case's Software Architecture

In general, the lower-level modules are executed at faster loop rates. The vehicle controller code executes at 20 Hz in hard real time. Moods are implemented as data-flow systems; they respond quickly to Observer events, but they do not require steady sampling or a hard real time implementation. For the various Observers, timing demands depend on their roles. For example, keeping track of the vehicle's physical state is performed at 20 Hz (to keep up with the vehicle controller), whereas an Intersection-is-clear Observer may respond with delays of seconds and still be adequate. The planner has the least demanding timing requirements. In the limit, planning may be performed as a batch pre-process, and thus planning would not have any real-time demands. However, the planner may be called upon to re-plan when a route is unexpectedly blocked, and planning results should be computed fast with respect to excessive wait times (e.g., a few seconds).

Our computing systems consist of: a low-level Compact RIO controller, three National Instruments PXI industrial computers (one of which runs the Farlap real-time operating system), three Mac-mini computers, and a dedicated "vision appliance." Our software is written primarily in LabView, although we also interface to C-code modules via DLL's.

Further details on our major subsystems are introduced next, in bottom-up order.

3. *Vehicle and Safety Systems*

3.1 DEXTER

Our vehicle, DEXTER, was commissioned by ENSCO, Inc. and built by Lothringer Racing on a stock Lothringer chassis. While DEXTER was designed for the Desert Challenge, it also meets all vehicle requirements for the Urban Challenge. It has excellent handling characteristics that allow it to operate autonomously at speeds up to 50 mph. The vehicle is propelled by a 2.0 liter supercharged ECO-Tech engine. A 5 kW alternator continually charges the 12 V and 48 V battery packs that provide power to DEXTER's electronics. DEXTER's adjustable suspension provides excellent stability and isolates the onboard computers and sensors from shock and vibration, yielding greatly reduced sensor noise and false obstacle detections. Due to its racing design, it is highly resistant to rollover and resilient in case of an impact. Furthermore, DEXTER's vehicle controls were designed specifically for autonomous operation and offer superior performance to retrofitted actuators on a standard automobile.

DEXTER's perfect run at the 2005 DARPA Grand Challenge National Qualifying Event (NQE) and performance in the final event testify to its ruggedness and competence. Further, as of 3/31/07, DEXTER had completed over 500 miles of autonomous operation. We expect that DEXTER is the only Urban Challenge vehicle to have achieved this milestone by this date. Upon completion of this project, DEXTER's capabilities will have been extended to meet the challenges of both urban and off-road environments.

DEXTER has a proven safety record and was designed with safety in mind. The fuel tank is a 32-gallon Fuel Safe racing cell that has a bladder and is foam-filled to prevent spill or an explosion. The electric fuel pump is directly controlled by the E-Stop Disable relay. If a system failure results in power loss, the spring-loaded emergency brake applies full braking. Front and rear OEM bumpers from similarly-weighted cars have been added. For testing and demonstration, we are able to enforce a limit on rpm's (and thus engine power) through the reprogrammable engine computer. A light-bar display at the rear of DEXTER displays status information that enhances safety. Displayed messages include: transmission gear, display of intent to proceed into an intersection, and reporting of lead-vehicle detection.

3.2 Vehicle Servo Controls

The Vehicle Controller (VC) incorporates throttle and braking control, similar to conventional cruise control, as well as steering control and reflexive safety monitoring. These functions are written in LabVIEW RealTime, they run on a National Instruments PXI-8176 Real-Time controller, and they execute at 20 Hz sample rate.

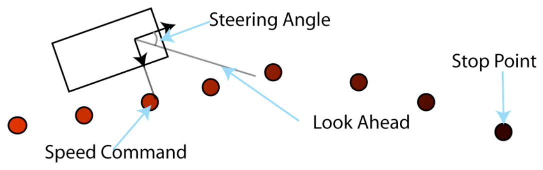


Figure 2: Illustration of how the Vehicle Controller determines speed and steering angle from the VBIL path points.

Throttle and braking control are performed with feedforward control in combination with a PID feedback control loop. Speed commands are profiled to achieve smooth accelerations and accurate braking at stop lines.

Steering control is performed by an NI-7344 Motion Controller to actuate the electrohydraulic steering mechanism. Steering angles are commanded using a "wagon handle" algorithm with respect to a defined path. DEXTER's current position and heading are compared to a speed-dependent "look-ahead" point on the path. A virtual wagon handle is computed to intersect the path. The length of the virtual wagon handle is modulated as a function of speed to maintain stability and avoid oversteering at higher speeds while achieving accurate path following at lower speeds.

3.3 Safety Interlocks

Safety interlocks in the Vehicle Controller are implemented as a state machine. States correspond directly to the transitions of the E-Stop signal, ensuring that DEXTER cannot operate the throttle unless the E-Stop is in Run mode. Also, the transition to shift requires the vehicle to be stopped in order to engage, making it safe to change into any gear and go into Pause mode. The remote E-stop system utilizes a heartbeat monitor, assuring that if communication is lost, DEXTER will shut down.

An additional safety interlock for computer-system health monitoring has been added. The VC monitors a UDP port for updated trajectory information from the higher levels. If the trajectory (breadcrumb path) fails to be refreshed within at most three seconds, the command is marked as "stale" and the VC will coerce DEXTER to a halt.

3.4 Safety Reflexes

An additional safety mechanism involves a reflexive filter that modulates commanded speeds constraining DEXTER to safe speed commands. Speed profiles are evaluated with respect to path curvature and acceleration/braking limits prior to execution to ensure safety and precise speed control. The reflexes also look ahead along the commanded trajectory to detect stop points and command speed tapering in advance of stopping. The reflexive speed profiling may override computed trajectories, as necessary to assure safety.

A safety reflex also monitors conditions of shifting vs speed and throttle. Shift commands are only executed if DEXTER is halted, and throttle commands are limited during shifting and when in park or neutral.

3.5 Vehicle/Behavior Interface Layer

The Vehicle-Behavior Interface Layer (VBIL), which is hierarchically between the Behaviors and the Vehicle Controller (VC), constitutes the interface through which higher levels of control can induce actions from DEXTER. The VBIL enforces safety constraints and ensures proper formatting of trajectories generated by the Behaviors. If a path is deemed partly infeasible, e.g., if a curve is requested that is tighter than DEXTER is capable of following at a given speed, the VBIL will throw an error and generate a speed profile along the trajectory in order to stop before the dangerous portion of the curve. Invalid and unsafe paths are rejected.

The VBIL performs safety checking at a higher level than that of the VC. Notably, vehicle following and queuing behavior is implemented as a filter in the VBIL. This module receives a lead vehicle's speed and position relative to DEXTER from the Lead Vehicle Observer. A hybrid controller modulates the path's velocity profile, if necessary, to accommodate for a lead vehicle and maintain a safe following distance. The minimum safe following distance is a function of the maximum of three measurements: a constant minimum distance (for queue stopped state), the DARPA-imposed one-vehicle-length per 10 mph, and the distance needed to safely come to a complete stop.

Collectively, the vehicle controller subsystem helps maintain safety through continuous error checking and reflexive responses. It runs on its own computer and it is accessible to higher levels only through a defined interface that incorporates error checking. The safety systems designed into the VC have supported safe and efficient development and testing of higher-level code.

4. Sensor Systems

DEXTER's sensors include both conventional sensors (GPS, inertial measurement, wheel encoders, steering angle encoder, and array of color cameras, a stereo vision system, Lidar units, vehicle radar) and some novel sensors under development (vision-enabled IMU, infrared cameras and ultra-wideband radar).

4.1 Physical State Sensors

A collection of rugged sensors has been chosen for sensing physical state. Hall-Effect wheel encoders placed on two of the robot's four wheels provide displacement measurements, while a steering encoder measures steering angle. A pair of GPS (Global Positioning System) receivers is used to establish DEXTER's physical state with an accuracy of 10 cm. The robot's initial heading is obtained via the relative location of two GPS units. A Novatel ProPack-LBplus, equipped with Omnistar HP and integrated with a Honeywell AG11 tactical grade LASER gyro IMU (Inertial Measurement Unit), has been selected to be the main source of GPS solution. The GPS receiver, loaded with Novatel's patented SPAN filtering system, provides IMU-only solutions to within 1^{o} per hour drift in the case of GPS outage.

4.2 Lidar, Radar and Vision Sensors

DEXTER is equipped with multiple SICK LMS-291 Light Detection and Ranging (Lidar) sensors mounted on the front, sides, and rear. Each Lidar unit delivers 180^{o} field of view, 80 m range scans with 0.5^{o} resolution at 40 Hz on a 500 kbps RS-422 serial bus to a National Instruments PXI computer. Distance values returned by the Lidar units are accurate and reliable. However, depending on the consistency of a surface and the angle at which it is struck, the Lidar scanners may fail to detect an object.

In order to detect objects that do not show up well on Lidar, an obstacle detection system based on the Point Grey Digiclops stereo camera is being integrated. A stereo camera system is more robust than the Lidar data in some instances, because it provides object detection capability that is not limited to a single height plane, as is the case with Lidar scanners. The Digiclops unit can accurately determine the distances to objects in its field of view out to 15 m, making it suitable for obstacle detection during low-speed driving.

In order to extend the range of our stereo capability, a new stereo camera system [Videre Design 07] with wider camera separation and a longer detection distance is being evaluated. In the case of both units, obstacle detection data will be merged with other sensing modalities to optimize sensing quality.

DEXTER is also equipped with an Eaton Vorad EVT-300 forward-looking Radar, primarily for the detection of moving vehicles ahead of DEXTER. The EVT-300 operates at 24.725 GHz and can track up to twelve obstacles concurrently within a 12^{o} Radar beam over a 100 m range. The Radar unit communicates directly to an Eaton VBOX interface device that in turn communicates with a PC over RS-232 serial at 19.2 kbps with information about tracked obstacles. A dynamic linked library (DLL) driver provided by Eaton handles all communication between the PC and the VBOX, and communicates with our software using function callbacks. These callbacks provide information about the obstacles that the Radar is currently tracking, as well as notices on when new objects are acquired and when old objects are released. The DLL communications and C calls are wrapped in a LabVIEW driver, which further processes the data from the Radar by providing a current list of tracked targets to a LabVIEW network data socket. Thus, the current Radar obstacle list is available to other processes on-demand.

Finally, DEXTER is outfitted with eight Imaging Source DFK21AF04 640×480 resolution color FireWire (IEEE 1394a) cameras. Each is set in a custom adjustable mount that protects the camera and shields it from glare. The cameras have been positioned and aimed with: 1) two wide angle lens pointed straight ahead, 2) one zoom lens pointed straight ahead, 3) two wide angle lenses angled diagonally forward to the right and the left, 4) two wide angle lenses pointed outward from each side, and 5) one wide angle lens at the rear aimed behind the robot. These mountings were chosen in order to provide complete visual coverage around the robot with additional coverage in front. The combination of wide angle and zoom lenses allows different regions to be recorded at spatial resolutions suitable for image processing while using the minimum possible pixel resolution in order to reduce computational time.

4.3 Ultra Wideband Radar

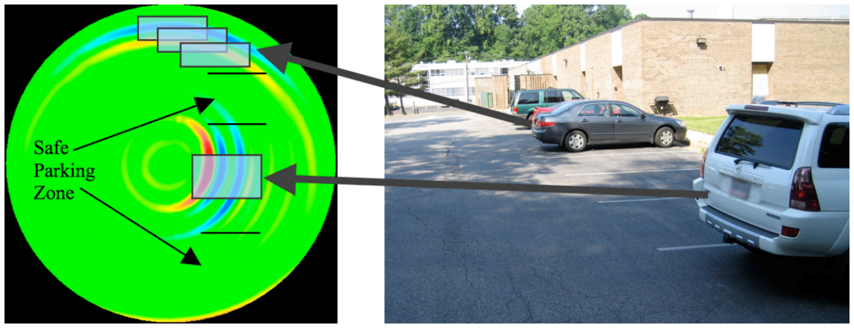


Figure 3: RAW scope view of UWB Radar for a parking lot scenario.

A novel sensor that we will employ is a custom-built ultra wideband (UWB) active Radar (being developed at ENSCO). Compared to existing sensor technology, UWB Radar systems have superior obstacle detection accuracy, operate in any weather conditions, and can penetrate through and detect concealed objects. Broadband electromagnetic pulses can easily penetrate through objects such as buildings and are immune to multipath cancellation effects seen in an urban environment [Multispectral Solutions 03]. A limitation of most such systems is that they are unidirectional, have a resolution of ±5 cm, and are unable to sweep a full 360º around a vehicle. Our new system can sweep 360º while maintaining a resolution of ±2 cm. Additional features have been added to improve reliability during adverse weather and road conditions.

Custom circuitry has been designed to address various problems associated with Radar systems when they are coupled to a vehicle platform. An internal 3-axis accelerometer and variable attenuator allow the system to filter out ground clutter from undesirable pitch and roll inherent to most vehicle platforms.

Fig. 3 depicts the raw Radar data and actual perspective of a typical parking lot scenario. At a range of 9 m the Radar can track multiple obstacles with an accuracy of 2 cm or better. This allows the system to distinguish closely spaced objects, obstacle characteristics, and even detect nonmetallic objects, unlike traditional narrowband Radar systems.

A distinct advantage of this UWB Radar system is its ability to penetrate through walls and other objects to maintain a lock on certain landmarks. Tracking these landmarks precisely is essential to the stability of our simultaneous localization and mapping (SLAM) system (see below). The system combines accelerometer and sweep position data of each trace to track moving obstacles precisely. It is also able to track objects precisely even when the unit is not moving, unlike Doppler Radar systems, which are prone to interference and have high false alarm rates [Fontana 02]. Mounted atop DEXTER, the Radar will have a largely unobstructed view in all directions.

4.4 Infrared Cameras

Team Case has acquired 3 uncooled pyroelectric-type infrared cameras for use on DEXTER. Infrared sensing will offer another sensing modality with its own unique capabilities. These sensors were only recently acquired and are still under evaluation. Promising opportunities include: sensing temperature differences between road surfaces and bounding foliage; identifying (hot) vehicles from background; sensing warm vehicle tires, radiators and exhausts; sensing the difference between a (cold) parked or stalled vehicle vs. a live vehicle; and detecting cooler fiducials, such as telephone poles, from background noise. We have observed that thermal imagery is also capable of detecting road markings, presumably due to differences in emissivity in painted regions.

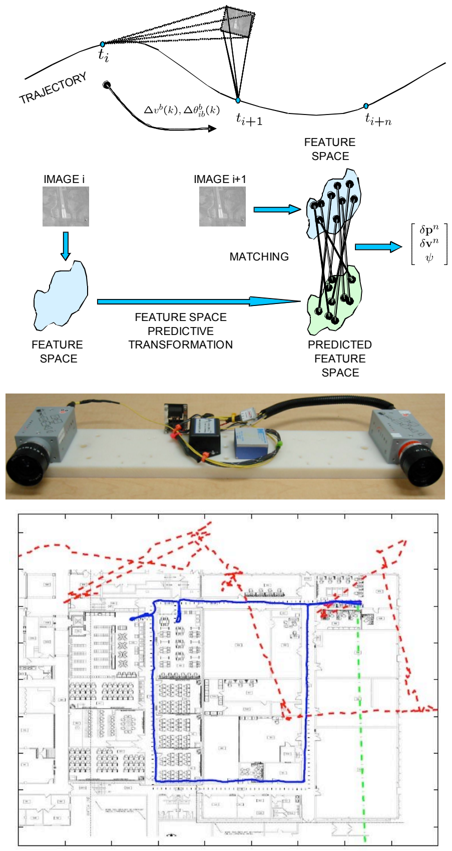
Information from our thermal cameras will be fused with other sensing systems to improve the robustness of sensory interpretation.

4.5 Fiducial-Aided Inertial Navigation and SLAM

In order for DEXTER to navigate under conditions of degraded or non-existent GPS signals, alternative navigation techniques that can meet GPS-level accuracy will be required. The Advanced Navigation Technology (ANT) Center at the Air Force Institute of Technology (AFIT), which is committed to researching alternative navigation techniques, is contributing a novel navigation system for Team Case's use.

The AFIT system integrates extraction and tracking of features from vision- and range-based sensors with IMU information. Localization drift is controlled by using Simultaneous Localization And Mapping (SLAM) techniques that rely on the same set of vision-extracted features. Currently, feature extraction is being done using the Scale-Invariant Feature Transform (SIFT) [Lowe 04], and feature association is performed stochastically.

The current implementation uses stereo vision (two Pixelink 1024×768 resolution grayscale cameras) and a low-cost Crista IMU. This system was used to collect data both indoors and outdoors that post-processed a navigation solution with an error growth on the order of one meter over several minutes of operation (Fig. 4). The Graphics Processing Unit (GPU) of the processing computer is used to perform the feature extraction, tracking, and navigation solution calculation in real-time [Fletcher et al. 07]. This system will be integrated with other navigation sensors and techniques to create a truly robust system. Integration of measurement sources will be accomplished using an Extended Kalman Filter (EKF).


TRAJECTORY
t_i
$\Delta v^b(k), \Delta\theta^b_{ib}(k)$
t_{i+1}
t_{i+n}
FEATURE SPACE
IMAGE i
IMAGE i+1
MATCHING
$\begin{bmatrix}\delta \mathbf{p}^n \\ \delta \mathbf{v}^n \\ \psi\end{bmatrix}$
FEATURE SPACE
FEATURE SPACE PREDICTIVE TRANSFORMATION
PREDICTED FEATURE SPACE
Figure 4: Use of feature tracking to correct IMU error, our prototype sensor, and data comparing the system's results (blue) with uncorrected solutions (red, green).

5. PERCEPTION

The process of converting sensory data into meaningful interpretations is perception. We are addressing the perception problem through a collection of specialized signal processing

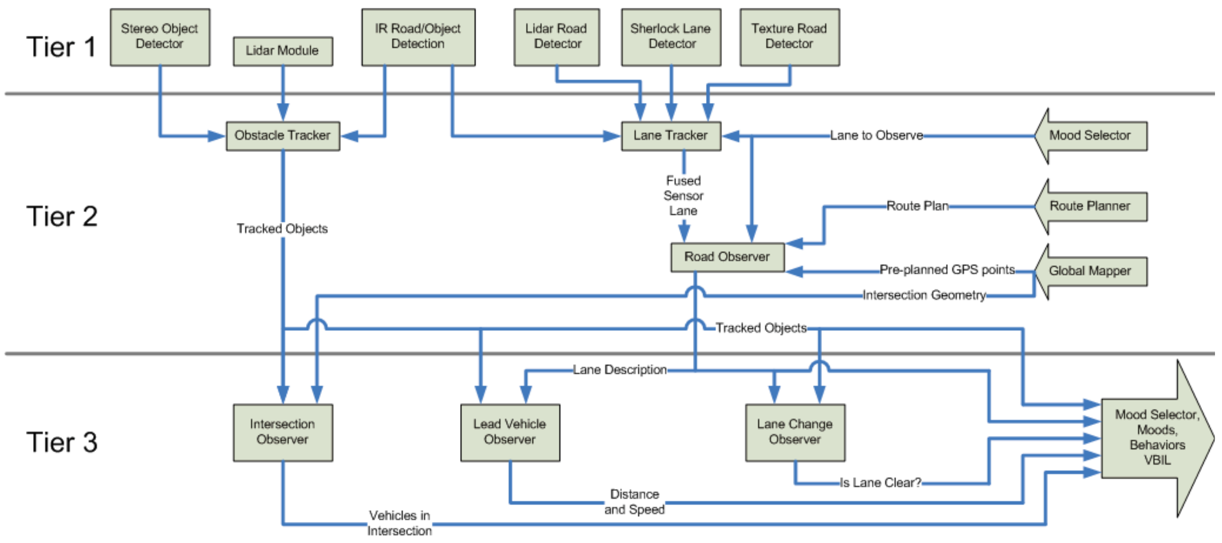


Figure 5: The three-tiered Observer architecture used to interpret DEXTER's environment. Data becomes increasingly abstract and decreases in volume as it moves through successive tiers.

processes that focus attention on key perceptual needs. In the spirit of the Luenberger observer, we refer to these generalized sensors as "observers," since they combine a priori and model expectations with sensory data to arrive at estimated properties. Three observers in particular are highlighted here: a Physical State Observer, a Lane Observer, a general Obstacle Observer, a Lead Vehicle Observer, and an Intersection Precedence Observer. An overview of the Observer system is illustrated in Fig. 5.

5.1 *Physical State Observer*

In mobile robotics, the physical state of a vehicle is vector of state variables (geographic location, velocity, yaw, pitch, and roll) that correspond to a dynamic time-dependant system. Our Physical State Observer (PSO) aims to optimize the accuracy of the physical state vector through the use of a kinematic model and multiple sensors. The PSO is perhaps one of the most important Observers as virtually all modules in our software architecture depend on it.

DEXTER is already equipped with the Novatel SPAN system, which integrates IMU and precision GPS information. The primary weakness of this system is IMU drift during GPS blockages. We are improving on the SPAN system through integration with a vehicle kinematic model and with the AFIT navigation system.

A kinematic model has been developed for DEXTER and calibrated through tests. Steering angle encoder values versus equivalent path curvatures have been calibrated using the SPAN system during tests at Goodyear's Vehicle Dynamics Area. While the kinematic model is not as good as DEXTER's IMU, the model can improve on the IMU solution by enforcing zero translation and rotation when the wheel encoders indicate DEXTER is at rest. Also, the kinematic model constraints heading rotation with respect to vehicle translation. Combining these virtues of the kinematic model with the SPAN system is done through an extended Kalman filter (EKF) [Gelb 74], which yields a state estimate that is superior to either system alone. We will also be integrating information from the AFIT system through an additional EKF term.

5.2 *Road Detection and Lane Tracking*

In the case where accurate and reliable GPS knowledge of a road and accurate knowledge of vehicle state are available, driving down a lane properly can be accomplished by a robot that is effectively blind. In all other cases (imprecise physical state estimate, imprecise a priori road coordinates or both), the robot must utilize additional sensory information. To maximize its ability to detect the road, DEXTER will utilize multiple sensors and sensor processing techniques. Each of these modules will detect different features of the road. A higher-level module, the Lane Tracker, will merge these sensory inputs with a map of the road layout to optimally estimate where individual lanes are located.

Roads are identified by many different types of boundaries, including lines, curbs, texture changes, color changes, cross-sectional profile, bordering foliage, curbs, and thermal signatures. Team Case is analyzing Lidar scans to identify which features are best detected by Lidar. Thermal features are being analyzed using infrared cameras. The dominant modality for roadbed detection is expected to be vision.

Our approach to lane detection is to utilize multiple sensors and multiple feature extractors and to pass these results on to a second-tier filtering Observer where they are combined into a unified perception. For vision processing, two different algorithms are being pursued in parallel. One of these uses a commercial machine vision package [DALSA IPD 07] to detect curb and line edges. The second is a combined color-texture classifier aimed at detecting the pavement surface. Each of these processing methods focuses on different cues in the image, leading to improved reliability when one cue is obscured.

Our edge feature extractor assumes the vehicle is facing a direction nearly parallel to one of the lanes of the road, and that image frames are taken from a forward-looking camera on the front of the vehicle. Using these assumptions, the image regions that must be analyzed are reduced to specific regions of interest. In these regions, points along the edge of the road or on lane lines are searched for using edge detection, and pattern matching within each frame. Interesting features found within the road frame are then fit to a model of the road, using a Random Sample Consensus (RANSAC) [Zhu et al. 06] algorithm and least squares fit of the data. The most successful fits, those containing the most features, are then assigned a confidence value according the number of features in the fit, and the location of those features within the image frame. Fig. 6 shows examples of successful road line fits. These successful fits, one on each side of DEXTER, are then passed to the Lane Tracker, which performs additional filtering on the results.

Color analysis and adaptive segmentation were used successfully in the previous DARPA Grand Challenge for long distance drivability analysis [Thrun et al. 06]. We are building on this approach with augmentation from texture analysis combined with color image analysis for short-range drivability evaluation. We are currently computing texture and its orientation in a manner similar to Zhang and Nagel [94] using the strength of the texture to segment road surfaces. We are also evaluating the use of Wavelet filters to compute similar measures [Rasmussen 04]. We will investigate adaptive learning algorithms to locally use color and texture to localize drivable surfaces.

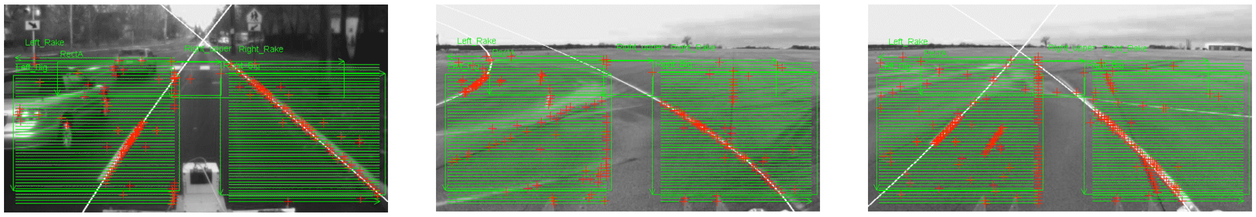


Figure 6: Images from the edge-based Road Detector show the many edge-detection points as well as the parabolic line fits resulting from the RANSAC fit algorithm.

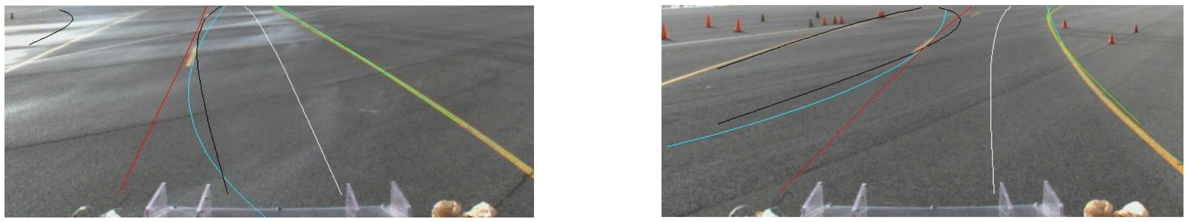


Figure 7: Images from the Lane Tracker Observer of the tracked left (red), right (green), and center (white) lines of the lane. Input road detection lines (blue), and other tracked lines (black) are not being used to calculate the centerline. In both images, the Lane Tracker is filtering poor road detection results.

The second level of filtering in DEXTER's road detection system is the Lane Tracker. It performs two functions: the first is to fuse the inputs from the various road detection sensor modules, and the second is to maintain a probabilistic model of the most likely road location as detected by the sensors. To combine sensory inputs, each contributor must report its estimate of lane coordinates along with an estimated variance. These contributions are merged with a model of the expected road location, which also has an estimated variance. These sources may then be combined with weights that are a function of the respective variances. Fig. 7 shows results from the Lane Tracker.

A good road model is an important component of a road detection system, and it should be one that can be quantitatively evaluated quickly and accurately models the road in all expected circumstances. (See [McCall, Trivedi 06] for a survey). The model being used by Team Case is a second-order polynomial fit. Our experiments have shown this model to be effective and computationally efficient. Additionally, we utilize model information from *a priori* knowledge and previous lane-observation experience, as stored within our Global Mapper module. The Mapper is the repository for innate and memorized information regarding high-resolution lane models.

5.3 Obstacle Detection and Tracking

Our obstacle detection strategy consists of separate obstacle detection routines run on each sensor, the results of which are fused to build a coherent obstacle map. This map is contained in the Obstacle Tracker module that is explained in detail further below. Since Lidar data provides the most accurate distance estimation towards other objects, it is the primary sensor used for obstacle detection. Other sensors are intended to increase confidence in objects detected by Lidar and to discern obstacles that the Lidar scanner was unable to detect.

Lidar scans are parsed using a custom LabVIEW driver. Following this, the data is smoothed with a 1^{o} half-width median filter and the resulting points are converted from polar to Cartesian coordinates. The Lidar scan data are then segmented into distinct line segments that are candidates for obstacle identities. These object locations are transformed into absolute GPS coordinates to test for persistence and credibility.

In our software architecture, the fusion of detected obstacles from each of the sensors occurs in the Tier-2 Obstacle Tracker Observer. The Obstacle Tracker also accumulates confidence in objects over time, maintains the existence of obstacles that have fallen out of sensor range, and calculates the speed and heading of moving objects. Condensing all of this information into a single data structure allows Tier-3 Observers to easily use it to make higher-level decisions. Essentially, the Obstacle Tracker is a sparse local map of the region around DEXTER stored in a GPS coordinate frame.

After converting inputs from the individual obstacle detectors into a common format, the newly detected obstacles are compared with the obstacles present in the previous iteration.

Matches are determined according to the distance and relative orientation of object pairs. Matches do not require an exact position match, allowing correct matching in the cases of noisy sensor reports and moving objects. Obstacles with repeated matches have their confidence score boosted and record both the previous and new positions.

Obstacles are maintained in memory unless their confidence score drops below a threshold or their distance from DEXTER exceeds a specified threshold. Obstacles that are not matched in the registration phase receive a penalty to their confidence score that will eventually drop them below the confidence threshold. Obstacle confidence scores are preserved if line of sight to them is obstructed by another obstacle or if they are outside the field of view of DEXTER's sensors.

By comparing the differences in position of successive locations of tracked obstacles (presumably other vehicles), a speed and heading for each of them can be determined. These velocities are used to predict future positions of obstacles. When an obstacle cannot be seen, its position is entirely dependent on these velocity predictions. When an obstacle is seen, the difference between its detected and predicted locations is used as a feedback term to modulate the velocity predictions.

5.4 Lead Vehicle Observer

As following a lead vehicle is a common and crucial task, a specialized Lead Vehicle Observer module has been implemented to ensure robust distance and speed measurements of a vehicle in the lane ahead. GPS coordinates for the present lane are parameterized as a function of the distance along the segment (lane odometry). An extended Kalman filter has been designed using this one-dimensional distance and its derivative, speed, as the state vector. The Obstacle Tracker returns Lidar-based position measurements of a lead vehicle in the current lane, and these coordinates are used as inputs to the Kalman filter. The resulting model provides reliable and consistent lead vehicle distance and speed estimates for use in vehicle following.

Fig.8 shows some quantitative results from the Lead Vehicle Observer. The data were obtained by logging DEXTER's GPS simultaneous with Lidar measurements, recorded along with a GPS time signal. During vehicle following, the lead vehicle also recorded high-precision GPS positions, also associated with the GPS time signal. The synchronized GPS recordings provided benchmark data for evaluating the Lead Vehicle Observer. As shown in Fig. 8, many of the Lidar readings—transformed to absolute GPS space—are very close to the true position of the lead vehicle. However, there are sections where the processed GPS position is inaccurate, potentially causing problems if the data were to be used as-is. The Kalman filter combines a lead-vehicle model with the Lidar data to obtain the lead-vehicle position and speed estimates.

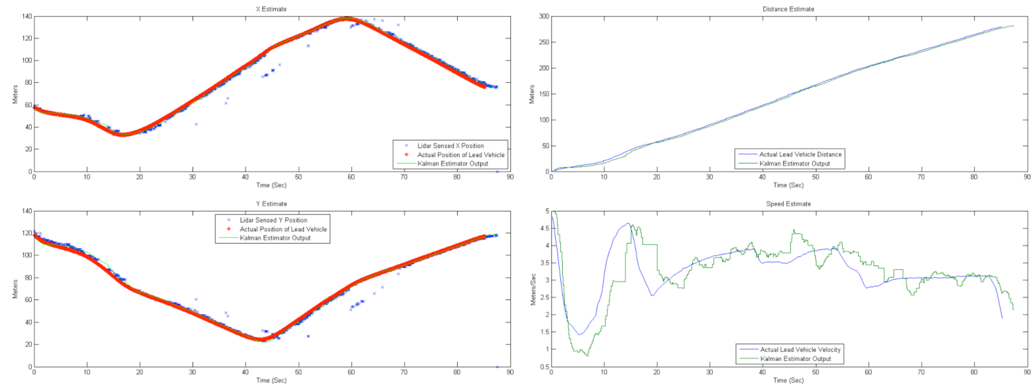


Figure 8: Demonstration of the Kalman Filter-based Lead Vehicle Observer with data logged from DEXTER and a lead vehicle to accurately estimate the vehicle's speed and position relative to DEXTER.

The resulting lead-vehicle state estimate is accurate and smooth, enabling good vehicle following.

The Kalman filter combines a lead-vehicle model with the Lidar data to obtain the lead-vehicle position and speed estimates. The resulting lead-vehicle state estimate is accurate and smooth, enabling good vehicle following.

5.5 *Intersection Precedence Observer*

Navigating through an intersection (with or without stopping) is a relatively complex and dangerous task. For this purpose, a dedicated Intersection Observer has been constructed (Fig. 9). After receiving notice from the Mood Selector that DEXTER is approaching an intersection, a UDP query sent to the Global Mapper returns a list of relevant intersection region estimations. The obstacle list published by the Obstacle Tracker is repeatedly read to build speed and heading profiles of objects near the intersection regions. Analysis of these profiles and the object positions can reveal when other vehicles have come to a stop, when they have entered the intersection, and when they have left it. The Observer then reports to the Mood Selector whether the intersection is clear (i.e., no moving traffic in it) and for how long cars at each of the entrances have been stopped.

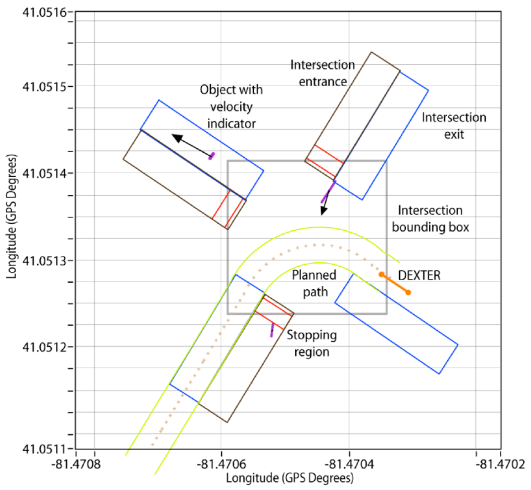


Figure 9: Intersection Observer showing regions used to help determine precedence and obstacle vehicles.

Tracking of vehicles at intersections is enhanced by replication of the Extended Kalman Filter construction used for the Lead Vehicle Observer—but applied to multiple vehicles moving along multiple candidate path segments.

In addition to refining state estimates for vehicles arriving at intersections, it is also necessary to assign precedence to each arriving vehicle. This requirement is made difficult by the ambiguity inherent in defining “arrival time” and “simultaneous.” Humans learn to classify arrival times and simultaneity from experience. For DEXTER to have similar competence, we are evaluating a neural-net classifier that is trained on actual records of vehicles approaching an intersection. A test vehicle recorded HP GPS while performing stops at stop lines. The example profiles were shifted relative to each other in time, and “correct” precedence was assigned by human judgment for each example.

This information was sorted into two groups: a training set and a validation set. The data was concatenated to express it as a single vector describing two synchronized distance measurements: sequential samples of the distance of vehicle 1 from its stop line, followed immediately by the corresponding samples of distance of vehicle 2 from its stop line.

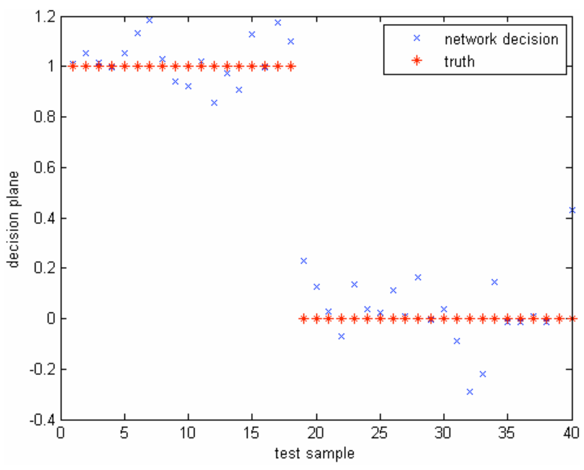


Figure 10. Results from neural network intersection precedence classifier.

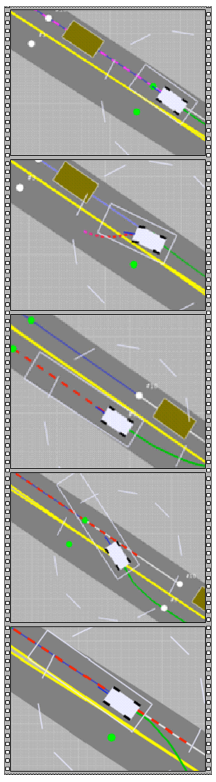
These relative arrival distance vectors, selected from the training set, were used to train a 3-layer, feedforward neural net using conventional backpropagation. The trained net was then evaluated on the validation set. It was found that the neural net successfully classified all precedences when the arrivals were staggered by more than 0.2 seconds. More nearly synchronous arrivals were harder to classify, thus appropriately triggering the "simultaneous arrival" rule.

Fig. 10 shows the results of the neural-net classifier for 40 validation examples. The network output was trained to output "1" if DEXTER had right of way and "0" if not. This network was run on the separate validation data, revealing that the network outputs would be correctly classified by a simple threshold of value 0.5. These results are promising for achieving precedence classification competence comparable to humans.

6. *Behavioral Control*

High-speed driving by autonomous vehicles typically has depended almost exclusively on a single behavior: a path follower. Most teams used this technique in the previous Grand Challenges. For the Urban Challenge, however, vehicles will require a variety of skills invoked reactively and appropriately in context. In our approach, we are modeling the way humans drive—not by computing every minute manipulation of vehicle controls, but by developing a set of distinct driving skills or Behaviors. Using subgoals prescribed by the Route Planner, along with situational awareness information from the Observers, our Behavioral Control scheme selects the most appropriate Behavior from an available library, within the context of the DEXTER's current situation.

Sensory-modulated trajectories from Behaviors obey dynamic and kinematic vehicle constraints and ensure appropriate margins between static obstacles and other vehicles. Behaviors are ordered into coherent maneuvers via context-enabled finite state automata called Moods. Each Mood automaton has as its states a subset of DEXTER's repertoire of Behaviors. The Mood Selector makes strategic decisions of how to sequence Moods to complete



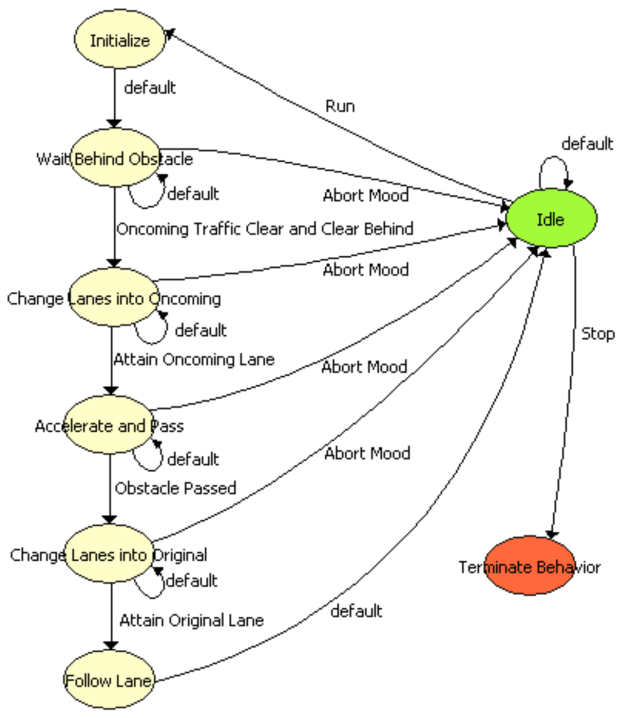


Figure 11: Demonstration of the Mood for passing a static obstacle. Upon coming to rest behind the obstacle (1), the automaton checks for oncoming traffic. Once clear, the behavior for changing lanes (2) is called and provides a smooth trajectory into the adjacent lane. The observed lane is switched by the Mood to observe the passing lane in the direction opposing traffic. The robot follows this lane (3) until the obstacle is completely cleared, at which point the lane description is switched back to the original lane (4) completing the maneuver (5).

a mission. Because the Mood Selector has constant situational awareness, it can switch Moods to best suit the situation, allowing for fast updates to situational changes and logical decision-making.

Any large AI problem requires careful segment-ation into digestible pieces. Identifying a granularity suitable for both rapid development and thorough testing is important to guarantee an algorithm will not only work as anticipated, but also yield the desired results. Incorporating the sensory-modulated trajectories of multiple Behaviors, Moods are equivalent to the skills learned by new drivers. The Mood Selector choreographs these skills together to accomplish prescribed missions. This architecture allows for rapid development, exhaustive testing in simulation and on DEXTER, and properly segments the autonomous driving task into a finite set of skills. Further details and illustrative examples follow.

6.1 Moods and Behaviors

In early development of DEXTER's control architecture, we determined that simple trajectory-generating Behaviors could not provide enough contextual information for an intelligent selection schema; nor would building a large, super automaton of such Behaviors be a logical solution. Our resulting architecture utilizes subsets of Behaviors organized into small, easily built and debugged automatons, each no larger than a handful of states.

Moods are independently executable, making them easy to develop and to repeatedly test, both in simulation and on DEXTER. Furthermore, additional Moods may be written with no effect to those already built. DEXTER's future capabilities are thus expandable by simply writing additional Moods. Fig. 11 demonstrates the Mood for passing a static obstacle.

Moods are easily created using LabVIEW's State Diagram Toolkit. State transitions may be managed with relatively little difficulty and are visualized in the state diagram editor (Fig. 11). Each Mood is able to query the Route Plan and Observers for current updates and information about the environment. Moods run asynchronously from the Mood Selector at 10 Hz and contain a set of Behavior member functions. When developing a Mood's capabilities, it may be run standalone, i.e., without the Mood Selector, in order to easily identify strengths and weaknesses of the code in different situations.

Our control architecture has seventeen Moods necessary for the DARPA Urban Challenge. Those that are fully developed allow the robot to follow a lane that is either clear or partially blocked, change into a lane of the same direction, stop at a stop point with queuing, proceed through an intersection once right-of-way has been attained, perform a U-Turn, and make turns from a major road onto a side street. Before the NQE, DEXTER's capabilities will be extended to pass moving obstacles on same-direction roads, negotiate parking lots and park, and drive through intersection traffic jams.

A Mood's state transitions use Observer data to actuate between the member Behaviors. One of the more complicated state transitions is triggered by the Intersection Observer. A transition from "Waiting" to an action of entering the intersection requires that the intersection is clear and DEXTER has right of way. DEXTER may also need to check for cross-traffic on roads without stop signs and that there is sufficient room in the destination lane such that DEXTER can fully exit the intersection. If these conditions are met, the robot proceeds into the intersection, completing the maneuver.

A second example of Mood-Behavior interaction is the U-Turn Mood. When a human driver performs a U-Turn, the approach taken is generally to follow the sharpest arc allowed by the vehicle until forced to stop by the edge of the road or an obstacle, shift into the opposite gear and turn the wheel to the opposite extreme, and to repeat this arcing until a direct path to the goal lane is achievable. The U-Turn algorithm developed for DEXTER borrows from this strategy.

When the Mood Selector indicates that a U-Turn is to be performed, the U-Turn Mood makes note of DEXTER's current position, current lane, and lane information (heading, position, width). This information is used to construct a legal bounding box that limits DEXTER's position during the maneuver to within the area allowed by DARPA specifications, avoids obstacles present in the road, and accounts for the position of the goal lane. A Behavior within the U-Turn Mood then generates a trajectory such that the robot pulls over to the right side of the current lane and comes to a stop, ready to begin the maneuver.

A planning algorithm then searches for a solution to the U-Turn problem by analyzing successive arcs at DEXTER's minimum turning radius. At samples along a candidate arc, the planner checks for the possibility of a direct path to the goal lane. If no immediate solution exists along the trial arc within the bounding-box constraint, the planner completes the arc, then reverses, steers to the opposite extreme and performs another arc search. This process is repeated until a solution is discovered. This process is also repeated (computationally) for each of the four possible initial moves, returning the shortest path to the goal, with preference given to paths with fewer direction changes.

Once a kinematic solution has been computed, a Behavior passes the first trajectory to the VBIL for execution. When DEXTER signals that it has stopped at the end of the arc, the robot's new pose is measured and a new complete solution to the goal lane is computed from the robot's updated position. This process is repeated until the trajectory to be executed is a direct path to the goal lane without direction changes. In this case, lane points are appended to the end of the trajectory and the U-Turn Mood gracefully exits, allowing a smooth transition back into the Follow Lane Mood.

6.2 Mood Selector

The Mood Selector prunes DEXTER's repertoire to only legal maneuvers given the lane markings (e.g., a double-yellow line), presence of obstacles, and location of other vehicles. Given a list of remaining legal maneuvers to perform, a strategic decision is made as to which Mood to execute. With clear road conditions and no legal restraints, following the present lane typically is the most strategic maneuver to execute. When a slow driver inhibits forward progress in the current lane, simply following a lane becomes less desirable. If there exists an adjacent lane, the strategic decision to pass a vehicle is a function of the feasibility of completing a pass before the end of the road and the adjacent road's maximum speed. Most decisions are mandated by the route network, as specified by the Route Planner, such as performing a U-turn or parking in a parking space. These maneuvers are high priority, although they must be interruptible by timeouts or unanticipated conflicts.

Only one Mood state machine is allowed to be active at any time, thus preventing competing trajectories from being sent by multiple active Behaviors. The Mood Selector protects against such conflicts, launches and aborts the Moods, and queries Observers to attain awareness of the environment and choose the most suitable automaton to achieve the next sub-goal. If a Mood is incapable of moving the robot forward safely, i.e., as it had been restricted by the Behavior, it will exhaust to an idle state and not be selected again until a suitable Mood is chosen from the remaining legal maneuvers. If no suitable Mood is capable of safely moving DEXTER in the forward direction, as in the case of an impassable roadblock or traffic backup, the Mood Selector signals the Route Planner to re-route and provide new directions toward the mission's next goal.

Additionally, the Mood Selector acts as the hub for information to be approved and published, e.g., the Route Plan and Global Map localization, to ensure consistent state information among AI modules. For example, upon approaching a stop line, the Mood Selector freezes the Route Plan until DEXTER has come to a full stop. The Route Plan is then forced to properly update to the waypoint through the intersection and the robot continues around the course.

7. Mapping and Planning

The highest levels of cognition in our system are the Route Planner and the Global Mapper. Each handles a different aspect of navigation: the Route Planner analyzes a connectivity graph between lanes to determine which ones DEXTER should drive along, while the Global Mapper stores precise GPS coordinates (and other spatially-tagged useful information) to help DEXTER travel down the lanes.

To be successful in navigating a complex urban environment with a coarse provided map, a vehicle can make use of its past experiences. History, both in terms of drivable area and static obstacles, can be leveraged to avoid repeating past mistakes (e.g., making an overly tight turn or attempting to drive around an obstacle covering multiple lanes). The Global Mapper addresses this challenge, and attempts to maintain a map of superior accuracy to that provided in the RNDF. This data is passed to the Observers in order to supplement current sensor data with past observations when possible.

The densely-connected and unpredictable nature of urban roads requires a frequently-updated high-level plan for traversing the environment. The Route Planner provides this plan, updating it when needed to account for changes in the vehicle's location and proximate goal. Additionally, the Route Planner includes descriptions of roadways and intersections to be encountered along the plan. The consumer of this data is the Mood Selector, which makes particular traffic decisions to follow the plan.

7.1 The Global Mapper

The Global Mapper is DEXTER's long-term memory. At the beginning of a mission, the Global Map contains all available *a priori* information from the RNDF and satellite imagery. As DEXTER moves through the environment, new information is selectively added. In this way, the Global Mapper supports learning in DEXTER. Other software modules can communicate with the Global Mapper via a set of UDP queries. These requests return information such as the GPS path of the upcoming lane and the entrance and exit locations of intersections.

Information contained in the Global Map is stored in "strings" of GPS "beads" that form a high-density representation of lane paths. These beads are originally determined by interpolating at one-meter intervals between the waypoints given in the RNDF file, but they can also be hand-edited in order to improve initial accuracy. In addition to GPS information, these beads store several other pieces of useful information, including a maximum velocity at the location, the total distance from the start of the lane, the heading of the lane at that point, and a measure of confidence in the accuracy of the point. We anticipate augmenting the bead data structures to hold additional useful information, such as GPS outages and easily recognizable fiducials. The Global Mapper sends out the contents of sequences of its beads in response to queries from DEXTER's road detection Observer.

In areas where *a priori* GPS descriptions of lanes are inaccurate, DEXTER's road detection system will cause him to drive down paths different from what is stored in the Global Mapper. As this happens, the Global Mapper will update its stored GPS beads to better match the path that was determined to actually lie on the road. The amount by which the beads are updated is governed by DEXTER's confidence in having an accurate GPS description of his location and whether he is currently driving in the lane (i.e., not performing behaviors such as obstacle avoidance.) Updated GPS beads will be used on successive trips down a lane, increasing the accuracy and reliability with which DEXTER can follow it. The beads stored in the Global Map can be saved and loaded to allow for preservation of memory between missions.

Whenever the locations of GPS beads stored in the Global Mapper are updated, the total shift performed on each of the beads is also stored. By analyzing overall shifts in the beads it is possible to detect and correct registration errors between the RNDF and the GPS coordinate frame. When the Global Mapper has sufficient confidence in a registration error, it can preemptively update unvisited beads in order to better estimate their locations.

In preparation for the support of advanced navigational techniques such as SLAM, the Global Mapper contains storage space with each bead to record the location of fiducials that could be recognized in successive lane traversals to aid in localization. In addition, if these fiducials can be recognized on a satellite map the Global Mapper could start with *a priori* knowledge of their location [Gonzalez, Stentz 07].

7.2 The Route Planner

The Route Planner's operation begins with the graph creation, which provides the graph data structure that is used in the Route Plan generation. In this context, each waypoint from the RNDF is a node in the graph, and each connection from the RNDF (e.g., between successive waypoints in a lane or between two waypoints in an exit) is an edge in the graph. Lane changing edges are also added in addition to the RNDF data to ensure reachability for goals that lie across lanes on roads with dotted white lines. Each edge is seeded with a weight based on an initial estimate of the time required to complete the journey between the two associated waypoints. Also, each edge is assigned a directive, which represents the action the vehicle should take between the two waypoints (e.g., turn left, stop, etc.).

The second task of the Route Planner is information extraction, which provides data from the RNDF that is not explicitly defined: the locations to enter and exit intersections and the organization of lanes within a roadway. This data is used to supplement the Route Plan by describing the set of lanes and stop locations that must be examined when analyzing precedence at an intersection and by exposing the directionality and markings of adjacent lanes for passing maneuvers.

The final Route Planner functionality is plan generation. As the only piece that is continuously running during the operation of the Route Planner, it is responsible for the majority of the Route Planner's computational load. Plan updating has two main components: determining the origin of the planning and executing the planning. To ensure continuous re-planning, the observation of a new origin triggers the generation of a new plan.

In order to provide an origin for the plan, the Route Planner samples DEXTER's physical state (GPS location and heading) and computes a weight function based on distance and heading deviation for each line segment in the RNDF (exits, sections of lanes, etc.) and outputs the waypoint pair (a forward waypoint and a backward waypoint) that realizes the largest weight. These computations are performed over the existing plan first, and the process is expanded to the full RNDF only if the results are below a threshold value of 0.75 (corresponding to a deviation of approximately 6 m and heading within 90º).

The localization is also responsible for determining when a goal (checkpoint) has been reached and for incrementing to the next goal. When the backward waypoint localization matches the previous forward waypoint localization, that waypoint is assumed to have been visited. When this occurs for the next MDF checkpoint, the goal is incremented. This process allows for the checkpoint to be counted if the vehicle passes over it directly or passes into an adjacent, opposing direction lane in the case of a stationary obstacle obstruction.

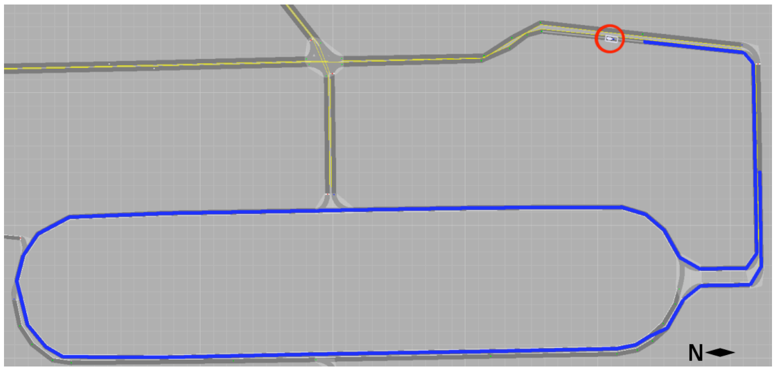


Figure 12: The planning process directs the vehicle around the traffic circle in the DARPA *Sample RNDF in order to reach Checkpoint #5 from segment 3's southbound lane.*

The planning process uses an implementation of Dijkstra's algorithm [Cormen et al. 01] for finding the shortest path in a graph. Using the forward waypoint localization as the origin, a plan is compiled that leads from the current location to the next MDF checkpoint. If the estimate of the time to complete this plan is less than a threshold time (30 seconds, assuring appreciable but not excessive length), the planning is repeated with later checkpoints until the total plan satisfies the condition. As a safety measure, the planning process always replaces the directive in the last element with a stop directive. A sample of the plan output is shown in Fig. 12.

The Route Planner is currently capable of dealing with complex, multi-lane routes and of fulfilling all of the requirements set out in DARPA's Basic Navigation and Basic Traffic Technical Evaluation Criteria [DARPA 07a]. This has been demonstrated through extensive testing with our Simulator and with on-vehicle testing, using over a dozen RNDFs of varying complexity. As the Route Planner evolves toward the functionality required for the Urban Challenge Race Event, there will be further development.

The coarseness of the information provided in the RNDF could lead to difficulty in localization for complex maps. Since the chief purpose of the Route Planner is planning, its ability to learn about the map should be limited to information relevant to planning (i.e., edge weights). However, the Global Mapper is responsible for learning the physical layout of the route, and therefore will have superior information for localization. As the map stored by the Global Mapper evolves during a mission, its ability to localize properly will improve. Therefore, the Global Mapper is better suited for localization on complex maps, and the Route Planner can be tasked solely with updating the Route Plan.

The chief aspect of Route Planner functionality that remains outstanding is the requirement for dynamic re-planning. While the Route Planner currently re-plans based on localization updates, in order to consider roadblocks it will receive and record notification of these blockages via the Mood Selector. As a result, the corresponding edge weights will be increased commensurate with the DEXTER's inability to traverse the edge, and later planning will tend to avoid the location. In order to "learn" the map from the Route Planner's perspective, it will also record the lengths of time spent driving along each of the graph edges, and utilize these time samples in an exponential weighted averaging scheme to update the planning data.

8. Simulation and Testing

Development and validation of DEXTER's software is difficult and inefficient to perform exclusively in the field. To address this barrier, we utilize three testing means: simulation, a road-legal instrumented vehicle, and field tests with DEXTER.

8.1 The Simulator

Our Simulator is a custom software package written in C++ that can run on any standard PC. Its primary purpose is to accurately model the aspects of vehicle behavior most pertinent to navigating traffic in an urban environment at relatively low speeds (30 mph or less), reflecting the evaluation criteria of the DARPA Urban Challenge. As such, we designed it as a mid-level simulation, eschewing the finer details of vehicle dynamics in favor of the ability to simulate dozens of vehicles at once.

DEXTER's AI is kept entirely separate from the Simulator, and it communicates only by the same means as used on the actual vehicle. Vehicle control commands are received by the Simulator instead of the actual car, and these commands are used to control the virtual test subject. The Simulator then computes approximations of actual sensor data and relays this information back to the AI. This leads to a vehicle-controller-in-the-loop type of testing, with the exact same control and decision-making processes in use on DEXTER (Fig. 1). As a result, we limit potential inaccuracies due to errors in vehicle dynamics and sensor simulation. Also, code tested on the simulator can be run verbatim on DEXTER, thus avoiding potential bugs introduced by translation, computing platform dependencies or recompilation.

The simulator uses the same vehicle dynamics model used by the Kalman filter for DEXTER's kinematic model. Additionally, the simulator models DEXTER's steering angle and steering rate limits, as well as acceleration and braking profiles and shifting delays. The appropriate values were measured directly from DEXTERor estimated by analyzing data extracted from experimental logs. Virtual physical state and Lidar information is derived from the simulation and follows the same processing paths as if it came from real sensors.

In order to exercise the test vehicle's traffic behavior, we created a set of agents capable of navigating in simulation. For navigation, they use a basic library of behaviors encompassing both good and bad driving, such as *drive in lane at speed*, *turn right at intersection*, and *stop dead*.

Agents are capable of navigating their own way around the simulation, serving as general traffic that interacts with DEXTER. Agents can last several hours without getting into "accidents" or "breaking laws," but they can be rigged to fail in ways similar to actual vehicles, such as stopping randomly, veering into oncoming traffic, or performing illegal U-turns.

The Simulator features a graphical display mode that overlays useful debugging information (trajectories generated by the Behaviors, the route plan, position history, tire tracks, etc.) directly on top of the map view, making it much easier to track down problems as they occur. As a result, the Simulator has become a primary testbed for all vehicle behavior and control not directly involving DEXTER's hardware.

Starting with our Simulator, we implemented a testing framework in which vehicle tests can be run and graded automatically. Using the scripting language Lua [Ierusalimschy et al. 96], we designed a set of performance evaluations that codify both the DARPA rules and more subjective judgments of vehicle behavior, such as smoothness of driving. These performance evaluations are capable of automatically verifying DEXTER's performance and reporting failures, and were used to implement two classes of tests: functional and randomized.

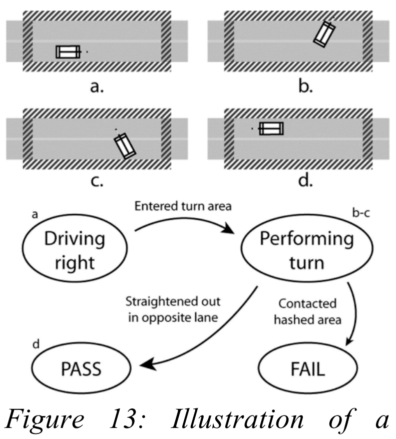


Figure 13: Illustration of a functional test showing *the state machine used to recognize legal U-turns.*

Functional tests are short, specialized scenarios designed to test the vehicle's conformance to a specific law or performance criteria. They are constructed by hand and use a combination of vehicle placement, obstacles, and agent behaviors to squeeze the test vehicle into a situation in which it must demonstrate a particular behavior, which is then recognized by a state machine. For example, Fig. 13 outlines the test setup for verifying proper U-turns. One law or rule frequently serves as the basis for several functional tests by changing the exact location and parameters and verifying that the rule is still obeyed.

Randomized tests are lengthy procedures in which the test vehicle is given a time-consuming (several hour long) mission to execute in a procedurally generated environment. Both randomly placed obstacles and rival cars populate the world, and DEXTER must survive in traffic long enough to complete its mission while obeying all performance evaluations. These procedures serve as stress tests, verifying that the vehicle can both remain stable in long missions and can properly handle the strange, unanticipated situations that inevitably arise during lengthy deployments.

Our testing system can automatically run a suite of functional tests against DEXTER's AI whenever changes are made to the code, generate a webpage detailing its conformance, and record logs to help developers track down where failures occur. In addition, we are establishing a test farm to continuously run randomized tests, to help identify unanticipated problems before they show up on the actual vehicle.

8.2 Road Testing with Didi

Since DEXTER is not road legal, we perform data acquisition and testing with an instrumented support vehicle, "Didi"—a 1997 Dodge Caravan. Didi duplicates most of the instrumentation available on DEXTER, including color and infrared cameras, Lidar units, odometry, steering angle sensing, HP GPS, and (most recently) an IMU. Didi also carries the same computing hardware (National Instruments PXI and Mac-minis) and runs nearly the same code as DEXTER. A significant difference from DEXTER is that steering and speed commands are not routed to computer-controls on Didi, but are instead displayed to a human operator. This puts the human in the control loop, enabling logging of DEXTER-like performance while preserving human-monitored safety. Didi is particularly valuable for evaluating algorithms concerning interactions with live traffic.

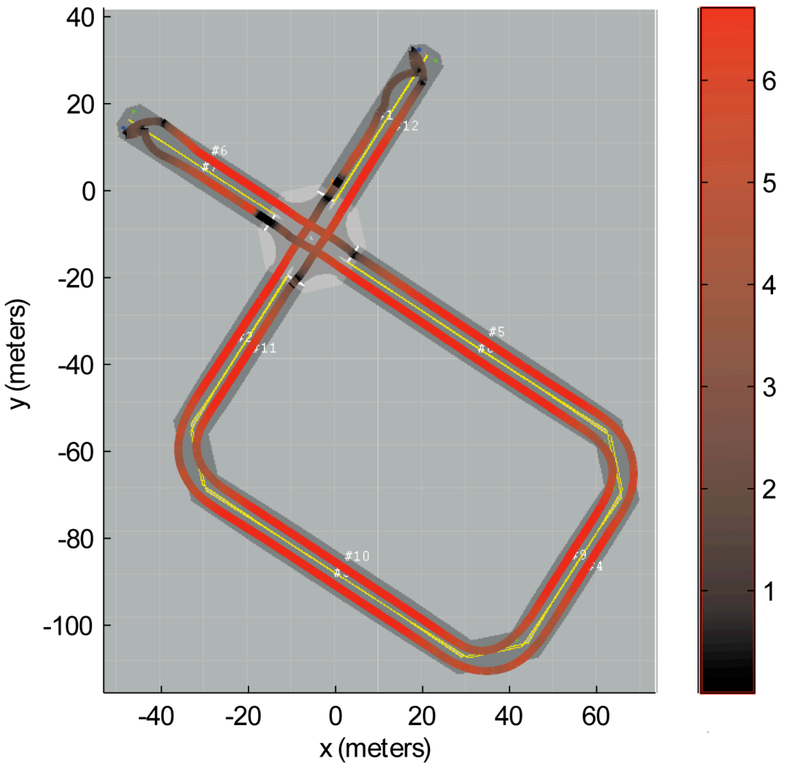


Figure 14: Spatial representation of DEXTER's logged velocity (m/s) during a mission performed on a copy of our Site Visit RNDF course at Goodyear's test track.

7.3 DEXTER Field Testing

Team Case has secured agreements for testing at 6 facilities: Case's farm, a large parking lot administered by Case, a large parking lot at Cleveland's International Exposition Center, the Vehicle Dynamics Area of Goodyear's Tire Test Track in Akron (Fig. 14), OH, and NASA's Plum Brook Station in Sandusky, OH. Using these facilities, DEXTER's software is tested regularly in field trials.

A valuable tool we have developed for testing is "DEXVIZ," a special visualization mode of our simulator in which we can "spy" on DEXTER's current state over a wireless connection and display the same debugging information available in simulation. When combined with a remote desktop application for monitoring logs and other output, we can identify and correct issues with the vehicle's software without ever touching DEXTER. The wireless router can be easily removed for competition purposes, and DEXTER will run on its internal software without any external dependencies.

9. *Conclusion*

This report has provided a technical review of Team Case's approach to autonomous robot construction. Crucial aspects of the design concern how to organize the software architecture such that it can support team development, incremental debugging and ongoing expansion. In making foundational choices, we have invoked inspiration from biology, including human judgment and human skills used in driving. Team Case is also advancing sensor and sensory processing technology in novel areas (e.g., UWB Radar and InfraRed sensing). Team Case will demonstrate DEXTER's competencies at its 6/22/07 DARPA site visit, and we are optimistic of advancing to the NQE and final race to pit DEXTER's performance against competing designs.

10. *References*

Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C., (2001) *Introduction to Algorithms*, 2nd ed. MIT Press and McGraw-Hill.

DALSA IPD, (2007) *http://www.goipd.com/products/Sherlock/*.

Defense Advanced Research Projects Agency (DARPA), (2007) "Urban Challenge Rules," DARPA Grand Challenge Website, *http://www.darpa.mil/grandchallenge/*.

Fletcher, J., Veth, M., Raquet, J., (2007) "Real Time Fusion of Image and Inertial Sensors for Navigation," *Proc. 63rd Institute of Navigation Annual Meeting*, Apr. 23–25.

Fontana, R.J., (2002) "Recent Applications of Ultra Wideband Radar and Communications Systems," *Ultra-Wideband, Short-Pulse Electromagnetics 5*, P.D. Smith, S.R. Cloude Eds., Springer.

Gelb, A. (Ed.), (1974) *Applied Optimal Estimation*, MIT Press: Cambridge, MA.

Gonzalez, J.P., Stentz, A., (2007) "Planning with Uncertainty in Position Using High-Resolution Maps," *Proc. IEEE International Conference on Robotics and Automation (ICRA'07),* pp. 1015–1022, Apr. 10–14, Rome Italy.

Ierusalimschy, R., de Figueiredo, L.H., Celes, W., (1996) "Lua – an extensible extension language," *Software: Practice & Experience 26*, No. 6, pp. 635–652.

Lowe, D.G., (2004) "Distinctive Image Features from Scale-Invariant Keypoints," *Int. J. Computer Vision*, Vol. 60, No. 2, pp. 91–110.

McCall, J., Trivedi, M., (2006) "Video-Based Lane Estimation and Tracking for Driver Assistance: Survey, System, and Evaluation," *IEEE Trans. Intell. Transp. Syst.*, Vol. 7, No. 1, Mar. 2006.

Multispectral Solutions, Inc., (2003) "Ultra Wideband (UWB) Frequently Asked Questions (FAQ)," *http://www.multispectral.com/UWBFAQ.html*.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nean, A., Mahoney, P., (2006) “Stanley: The Robot That Won The DARPA Grand Challenge,” *J. Field Robotics*, Vol. 23, No. 9, pp. 661–692.

Videre Design, (2007) http://www.videredesign.com.

Zhang, J., Nagel, H.-H., (1994) “Texture-Based Segmentation of Road Images,” *Proc. Intelligent Vehicles ‘94 Symposium*, Oct. 24–26, pp. 260–265.

Zhu, W., Chen, Q., Wang, H., (2006) “Lane Detection in Some Complex Conditions,” *Proc. IEEE/RSJ Conference on Intelligent Robots and Systems*, pp. 117–122.